info502 : Système d'exploitation
TP 1 : introduction à Unix, scripts

Pierre Hyvernat, Sylvie Ramasso, Brice Videau
Pierre.Hyvernat@univ-savoie.fr
Sylvie.Ramasso@univ-savoie.fr
Brice.Videau@univ-savoie.fr

Pour ce premier TP, pas besoin de rendre de compte rendu : il n'a pour but que de vous familiariser avec le système Linux. Essayez de tout faire, et n'hésitez pas à poser des questions ou à chercher des compléments d'information sur internet.

Exercice 0 : Démarrage des ordinateurs, Linux

Redémarrez votre ordinateur en choisissant le système Linux. Loggez-vous en utilisant votre login et mot de passe habituel.

Question 1. Passez une dizaine de minutes à explorer l'environnement de travail. Cherchez vos fichiers personnels et essayer de changer des options de configuration de votre compte (fond d'écran, préférences internet etc.)

Question 2. Allez lire la page wikipedia (`fr.wikipedia.org`) sur Linux. Qu'en pensez-vous ?

Exercice 1 : Prise en main du terminal, commandes usuelles

La plupart des utilitaires sont accessible depuis le "shell". Pour la suite du TP, lancer l'application "terminal" depuis les menus. Voici une petite liste des commandes importantes :

- `man` : c'est une application qui permet d'obtenir le manuel d'un programme particulier. Par exemple, "`man ls`" vous donnera le manuel d'utilisation de la commande `ls`.
- `ls` : cette commande affiche la liste de tous les fichiers, répertoires et autres dans le répertoire courant. N'hésitez pas à vous familiariser avec les arguments de cette commande (dont "`ls -l`")
- `cd` : cette commande permet de changer de répertoire courant (`cd` = "change directory").
- `rm` : pour effacer un fichier.
- `rmdir` : pour effacer un répertoire.
- `mkdir` : pour créer un répertoire.
- `pwd` : permet d'afficher le chemin du répertoire courant. (`pwd` = "print working directory")
- `chmod` : permet la gestion des droits sur les fichiers.

Remarque : Certaines commandes (comme `cd` ou `pwd`) ne sont pas de véritables programmes (elles ne correspondent pas à des fichiers exécutables). Pour obtenir l'aide de ces commandes, il faut faire "`man bash-builtins`".

Question 1. Quelle commande utilisez-vous pour obtenir le manuel de la commande `man` ? Pour avoir l'aide de l'*affichage* des pages de manuel, il faut faire un "`man less`".

Question 2. Utiliser la commande `pwd` pour savoir dans quel répertoire vous vous trouvez. Notez le résultat.

Déplacez-vous dans le répertoire `/bin/`. Est-ce que ce répertoire contient un fichier appelé `ls` ? Est- ce qu'il contient un fichier appelé `cd` ? Qu'en pensez-vous ?

Retournez dans le répertoire précédent. Une manière rapide de retourner dans l'ancien répertoire courant est d'utiliser la commande "`cd -`". Essayez...

Retourner dans votre répertoire personnel. (Une manière rapide de rejoindre votre répertoire personnel est de taper la commande "`cd`", sans aucun nom...)

Question 3. Créez un répertoire qui s'appelle "`TP systeme`". Est-ce que ça marche ?

Exercice 2 : Chemins d'accès

La plupart des programmes se trouvent dans l'un des répertoires suivants :
- `/bin/`
- `/sbin/`
- `/usr/bin/`
- `/usr/sbin/`

Lorsque vous tapez un nom de programme, le système vérifie que le programme existe dans un de ces répertoires. Si le fichier existe, il est exécuté. Sinon, un message d'erreur apparaît.

Remarque : le "Filesystem Hierarchy Standard" (FHS : `www.pathname.com/fhs/pub/fhs-2.3.html`) décrit les conventions de nommage et l'organisation des différents répertoires et fichiers systèmes dans les systèmes de la famille Unix. Reportez vous à la documentation concernant le : `/bin` ; `/sbin/` ; `/lib/` ; `/usr/bin/` ; `/usr/sbin/` ; `/usr/lib/` ; `/dev/` ; `/proc/` ; `/etc/` ; `/tmp/` ; `/var/` et n'hésitez pas à parcourir ces répertoires afin d'observer les fichiers qu'ils contiennent.

Pour exécuter un programme qui n'est pas dans un de ces répertoire, il faut donner son chemin d'accès. Par exemple, la commande `/usr/local/bin/demineur` exécutera le programme `demineur` du répertoire `/usr/local/bin/` (si le fichier correspondant existe).

Pour indiquer le chemin d'un fichier, on peut utiliser soit un chemin absolu (par exemple `/sbin/ifconfig`) soit un chemin relatif (par rapport au répertoire courant : "`./`" permet de parler du répertoire courant, et "`../`" permet de parler du répertoire père.)

Remarque : la touche Tab permet, si cela est possible, de demander au système de compléter un chemin d'accès incomplet. Par exemple, au lieu de taper "`cd /root/ABCDEFGHIJK`", vous pouvez taper "`cd /root/AB`" puis Tab. Le système rajoutera "`CDEFGHIJK`"

Question 1. Expérimentez avec la touche Tab. (En créant des répertoire et des fichiers avec des grand noms...). Que se passe-t'il si vous avez un répertoire "`ABCD`" et un répertoire "`ABEF`" ?

Question 2. Copiez le fichier `/bin/ls` dans votre répertoire "`TP systeme`" en lui donnant le nouveau nom `LS`.

Comment faire pour exécuter le programme "`LS`" ?

Question 3. Si vous êtes dans le répertoire "`TP systeme`", quel sera l'effet de la commande `.././TP\ systeme/./././LS` ?

Exercice 3 : Droits d'accès

Le système de fichiers utilisé (EXT2 ou EXT3) par Linux permet une gestion assez fine des droits d'accès aux fichiers. Pour chaque fichier, on peut restreindre les actions possibles : on peut avoir le droit de
- lecture (pour visualiser le fichier),
- écriture (pour modifier le fichier,
- exécution (pour l'exécuter),
- une combinaison arbitraire de ces droits.

De plus, on peut donner/supprimer ces droits pour :
- le propriétaire du fichier,
- le groupe du fichier,
- tous les utilisateurs.

Remarque : comme vos répertoires personnels n'utilisent pas le système de fichier de Linux, vous devez tester cette partie dans un autre répertoire. Je vous conseille le répertoire `/tmp/`, dans lequel vous créerez un répertoire `TP-systeme`.

Question 1. Créer un fichier vide avec un nom que vous aurez choisi en utilisant la commande "`touch nom_fichier`". A quel groupe appartient il ?

Question 2. Lisez le manuel de la commande `chmod` et changer les droits de ce fichier.

Question 3. Supprimez les droits de lecture sur votre fichier. Essayer de l'ouvrir avec un éditeur de texte. Remettez les droits de lecture, mais supprimez les droits d'écriture. Essayez de l'ouvrir avec un éditeur de texte et rajouter une ligne au fichier et essayez de le sauvegarder.

Question 4. À quoi correspondent les droit de lecture / écriture / exécution pour les répertoires ? Testez en décrivant ce que vous faites.

À quoi peuvent servir les droits d'écriture et exécution sur un répertoire si on n'a pas le droit de lecture ?

Sans droit d'écriture, on ne peut pas supprimer de fichier du répertoire ; mais est-ce que le contenu du répertoire est en sécurité ? Essayez.

Exercice 4 : Exécution avancée de commandes

Lorsque vous lancez un programme à partir du terminal, le terminal est bloqué jusqu'à ce que le programme en question se termine. Il est possible de dire au programme de s'exécuter "en arrière plan", c'est à dire que le terminal continuera de fonctionner pendant l'exécution du programme. La syntaxe est : "`commande &`".

Question 1. Essayez les commandes "`sleep 4`" et "`sleep 4 &`". Quelle différence constatez-vous ?

(La commande "`sleep`" ne fait rien d'autre que d'attendre un certain nombre de secondes avant de terminer.)

Question 2. Dans le second cas, vous ne savez pas vraiment quand la commande se termine. Essayer les commandes "`(sleep 10 ; echo "FIN")`" et "`(sleep 10 ; echo "FIN") &`". Que constatez-vous ?

(La commande "`echo "FIN"`" permet d'afficher "`FIN`" sur l'écran, et le point virgule "` ;`" permet de séquentialiser des commandes...)

Question 3. Lancer plusieurs commandes en arrière plan. (Par exemple, une commande qui attend 25 secondes avant d'afficher "`FIN 25 secondes`", un autre qui attends 1 minute avant d'afficher "`une minute, c'est long !`".)

Vous pouvez obtenir la liste des programmes qui s'exécutent dans le terminal en utilisant la commande "`jobs`". Essayez.

Comment interprétez le résultat ?

Question 4. Pendant qu'un programme s'exécute en premier plan (pas de "`&`"), il est possible de l'*arrêter* en appuyant (en même temps) sur "Control" et "`c`". On peut *suspendre* un programme en appuyant sur "Control" et "`z`".

Lancez les commandes

- `(sleep 60 ; echo "PREMIER")`
- Control-`c`
- `(sleep 60 ; echo "DEUXIEME")`
- Control-`z`
- `(sleep 60 ; echo "TROISIEME") &`
- `jobs`

Comment interprétez-vous le résultat de la commande "`jobs`" ?

Question 5. Si on a la liste des processus lancés à partir du terminal (commande "`jobs`"), il est possible de passer un programme en arrière plan avec la commande "`bg %n`" où `n` est le numéro du processus dans la liste. (`bg` = background) De la même manière, il est possible de passer un processus en premier plan avec la commande "`fg %n`".

Testez...

Question 6. Essayez de comprendre ce que fait la commande "`exec nom_de_commande`".

Exercice 5 : Variables d'environnement

Lorsque le shell s'exécute pour la première fois, de nombreuses variables d'environnement sont créées. Ces variables d'environnement contiennent des informations sur le système, l'utilisateur ou les programmes. Habituellement, les variables ne contiennent que des majuscules et le symbole "_" ; pour les afficher à l'écran, il faut utiliser "`echo $NOM_DE_VARIABLE`" (ne pas oublier le dollar) et pour les modifier, il faut utiliser "`NOM_DE_VARIABLE = ...`" (sans dollar).

Voici quelques exemples de variables d'environnement

- `HOME` : contient le chemin d'accès de votre répertoire personnel,
- `UID` : contient votre numéro d'utilisateur,
- `USER` : contient votre nom de login,
- `LINES` (et `COLUMNS`) : si vous êtes dans un terminal, `LINES` contient le nombre de lignes de votre terminal,
- `OLDPWD` : contient le chemin d'accès de votre précédent répertoire
- `PS1` : contient une description du prompt : ce qui précède chaque commande lorsque vous êtes dans un terminal
- `PATH` : contient la liste des chemins d'accès des programmes. Comme le chemin "`/bin/`" est dans `PATH`, vous n'avez pas besoin de taper "`/bin/ls`" mais vous pouvez vous contenter de taper "`ls`".

D'autres variables se rapportent au programme que l'on vient d'exécuter :

- `?` : contient la valeur de retour du programme précédent. (En général, c'est 0 si le programme c'est exécuté correctement, et une autre valeur sinon.)
- `$` : contient le numéro de processus du processus courant.

Question 1. Quel est votre numéro d'utilisateur ? Quel est celui de votre voisin ?

Question 2. Regarder la valeur de `LINES` et `COLUMNS` ; changer la taille de la fenêtre du terminal et recommencez.

Question 3. Rajouter le chemin du répertoire contenant `LS` à `PATH`. Essayer d'exécuter `LS` d'un répertoire quelconque.

Question 4. La commande "`cd -`" permet de retourner dans le répertoire courant précédent. C'est en fait un raccourci qui veut dire "`cd $OLDPWD`". Modifier la valeur de `OLDPWD` et utilisez la commande "`cd -`".

Question 5. Afficher le numéro de processus du shell dans lequel vous êtes ; vérifiez que c'est le bon en utilisant la commande `ps`.

Question 6. En allant voir la partie pertinente du manuel de bash, modifier la valeur de `PS1` pour faire un concours de prompt...

Exercice 6 : Un premier script

Un "script" est une espèce de petit programme effectuant des taches simples, le plus souvent dans un langage interprété. Un script shell est écrit en utilisant les commandes du shell (bash dans notre cas).

Utilisez un éditeur de texte pour créer un fichier contenant

```
#!/bin/bash
if (test -d "$HOME/Systeme-TP-1")
   then echo "le répertoire existe déjà"
    else echo "création du répertoire"
         mkdir Systeme-TP-1
fi
```

Question 1. Essayer de deviner ce que ce script va faire.

Question 2. Comment pouvez-vous exécuter le script ? Essayer. Est-ce que le résultat correspond à vos attentes ?

Question 3. Modifiez le script pour que sa valeur de retour soit 0 quand le répertoire est crée et 1 quand le répertoire existait déjà. (Pour retourner la valeur n, il faut utiliser “`exit` n”.) Testez et vérifiez que ça marche...